Università degli Studi di Milano

Machine Learning Project

Rock-Paper-Scissors CNN classification project

A Deep Learning Approach to Hand Gesture Recognition

Student: Gloria Villa

Student ID: 49117A

Academic Year: 2025/2026

February 2026

Contents

1 Introduction

Image classification is one of the most studied applications of deep learning these days, and CNN (Convolutional Neural Networks) is widely accepted as the standard architecture for this task. This is because a Convolutional Neural Network is a type of deep learning algorithm specifically designed for processing structural data like, in our case, images (but also videos and time series data). In fact, CNNs are particularly efficient at recognizing features and recurring patterns in visual datasets, such as the presence of an object in the images or identifying other anomalies in a data stream. In this project, three different CNN architectures of incremental complexity are applied to solve the classification task of the Rock-Paper-Scissors game to accurately classify images of the hand gestures present in the dataset.

The dataset used for this project was created as a part of a Raspberry Pi computer vision project as stated by the author of the kaggle page [1]. It consists of 2188 RGB images of size 300×200 pixels divided into three classes: - Rock: 726 images - Paper: 712 images - Scissors: 750 images All of these images are taken with a green consistent background under similar or even identical lighting exposure conditions.

The primary objective of this project is to design, train and compare the results given by three CNN architectures of incremental complexity: A baseline model, an Intermediate model and an Advanced one. The methodology used incorporates a nested cross validation with hyperparameter tuning and a fully automatic grid search to get the best configuration for the second CNN model (Intermediate CNN). In this case, hyperparameter tuning was applied only for the second model to maintain an acceptable computational runtime.

In the end a generalization test on the models was performed by using personally captured images of the Rock-Paper-Scissors hand gestures to see how the models would react on real-world images with a different background, different colors and different lighting exposures and background conditions. This also serves to investigate potential overfitting to dataset specific conditions such as the uniform green background present in all the images of the original dataset.

Note: I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.

2 Methodology

First of all, before any data processing, an Exploratory Data Analysis (EDA) was performed on the dataset to understand how the dataset is structured and also to find out more about the properties of the images of the hand gestures. This analysis was conducted through the use of different plots for showing class distribution, pixel intensity, brightness of the images, contrast and RGB channel distributions. The aim was to identify any processing requirements before going on with any training of the models on the dataset images.

Then, the dataset was split into training, validation and test sets to prepare the data for processing and training. After the splitting process, the images were resized and normalization and data augmentation were applied only on the training set since no data manipulation should be applied on the test set. Data augmentation, in particular, consisted of techniques like random flipping, rotation and zoom to improve the generalization of the models since it introduces more variability in the images.

When both EDA and data processing were completed, three CNN architectures of incremental complexity were created. The first was a simple Baseline model with two convolutional blocks, then complexity was increased with the Intermediate model where a third convolutional block was added along with dropout layer and a third model (Advanced) where 4 total convolutional block were implemented with the introduction also of Batch Normalization and Global Average Pooling.

The model evaluation was carried out using a nested cross validation of 5 folds across all three models, combined with the fully automatic grid search for the hyperparameter tuning applied only on the second model for computational runtime purposes.

In the end, a generalization test was conducted using images personally captured to verify the performance of the models on real-world images that introduce different backgrounds and lighting exposures.

3 Exploratory Data Analysis (EDA)

Before starting the training and test of the models, an Exploratory Data Analysis (EDA) was conducted on the raw dataset to examine the structure and the properties of the images data and have a better general knowledge of the data we are going to use for the classification task.

First of all, the distribution of the images among the three classes was checked and visualized through a bar plot. The dataset contains 726 Rock images, 712 Paper images and 750 Scissors images, for a total of 2188 images in the whole dataset. The bar chart created for showing the distribution among the three classes confirms that the dataset is approximately balanced across all the three classes (see Figure 1), with really a small difference in the amount of images per class. This was an important finding since it means that no class reweighting was needed before training.

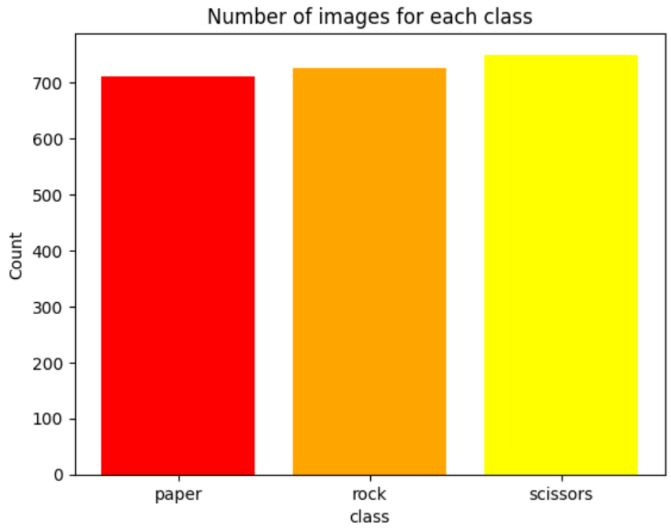

Figure 1: Class distribution of the Rock-Paper-Scissors dataset

Then, a random sample of six images was displayed for each class to perform a qualitative check on images consistency and label correctness (see Appendix 9, 10, 11).

Next, the properties of the images were verified across all the three classes. The images shared consistent dimensions of 300×200 pixels and they are stored as RGB images with three channels (Red, Green and Blue), and have pixel values that are in the range $[0, 255]$. So, all of these proved that no format conversion or dimension issues exist at the raw data level; resizing will be needed during the preprocessing phase of the images before going on with the training phase.

A pixel intensity analysis was then conducted on the images by plotting the distribution of all the pixel values as histograms (see Appendix 12, 13, 14). This analysis was conducted to see if any brightness or contrast corrections would be needed before training and also to verify if the images are well-exposed. All the images showed a consistent bimodal distribution, reflecting two dominant pixel groups present in the images:

- Green background (darker pixels)
- Skin tones of the hand gestures (lighter pixels).

The absence of extreme values at both ends of the intensity range of the pixels confirmed that the images are really well-exposed and no additional brightness or contrast corrections were required in the processing stage of the analysis before the training phase.

A brightness and contrast analysis was then performed on a sample of 50 randomly selected images per class to further verify the visual consistency of the dataset. The mean pixel value was used as an indicator of brightness and the standard deviation of the pixel values was used as an indicator for the contrast. The resulting values from this analysis were nearly identical across all the three classes (see Appendix 15), confirming the previous assumption that no brightness or contrast corrections were needed. A scatterplot of brightness vs contrast (see Appendix 16) further confirmed this finding, showing significant overlap between the three classes in the brightness and contrast space. All of this indicates that the CNN models that are going to be implemented in the next steps will not be able to distinguish hand gestures based on lighting differences alone and must instead learn structural and shape-based features from the images.

Finally, as the last step of the EDA phase, an RGB channel analysis was conducted on a sample of 20 images per class to examine the color distribution patterns of the dataset images. To have a better understanding of the distribution patterns of the colors in the images, the average values of Red, Green and Blue channels were computed and visualized through a bar chart (see Figure 2).

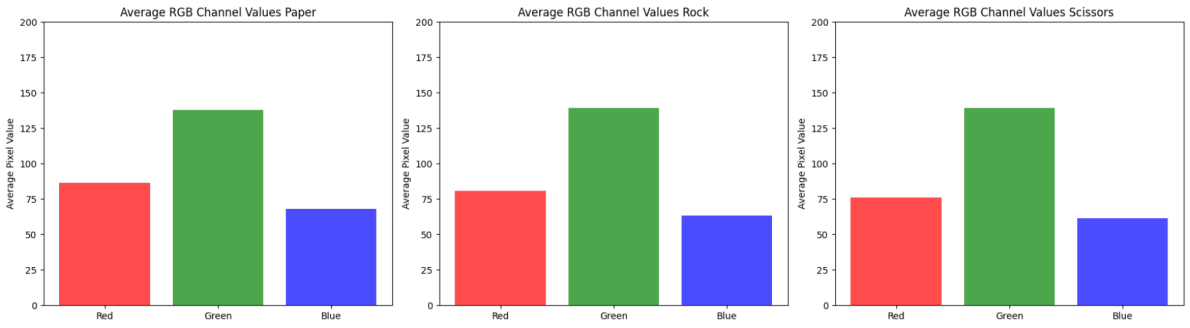

Figure 2: Average RGB channel values for each class.

The green channel was found to be the most prevalent one across all the three classes, justified by the fact that most part of all the images is occupied by the green background used for taking the photos of the hand gestures of the rock-paper-scissors game. This finding demonstrated that there is also no color bias, which reinforced the previous statement that the CNN models we are going to implement must rely on shape and structural features rather than color and lighting exposure to perform the classification task.

4 Data Processing

Before starting with the data processing part, the dataset was split into train (70%), validation (15%) and test (15%) sets at the file level to prevent data leakage between the sets, so that no information from the validation or test sets could influence the training phase. A fixed random seed of 42 was used to guarantee full reproducibility across sessions. To verify that no overlap is present between the three sets, intersection checks were performed (zero overlap was confirmed). All through the processing and evaluation phase,

the test set was kept completely untouched since no data manipulation should happen on the test set.

After the splitting, the images were loaded and resized to 64×64 pixels. The choice of the 64×64 for image size was made to reduce the computational cost of training while still retaining enough spatial information for the models to learn the hand gesture features. Other image sizes were tested during the development phase, like 150×150 or 28×28, but 64×64 was found to be the best configuration, balancing computational efficiency and model performance.

Then, pixel values were normalized, bringing them from the original range of $[0, 255]$ to the $[0, 1]$ range using a Normalization layer fitted exclusively on the training set. This was an important step since working with normalized images helps the models train more smoothly and reach convergence faster.

At the ending phase of the processing of the images, one last step was required: Data augmentation. Data augmentation was applied exclusively to the training set, like all other processing steps, to artificially increase the diversity within the images of the dataset and to reduce the risk of overfitting (see Figure 3).

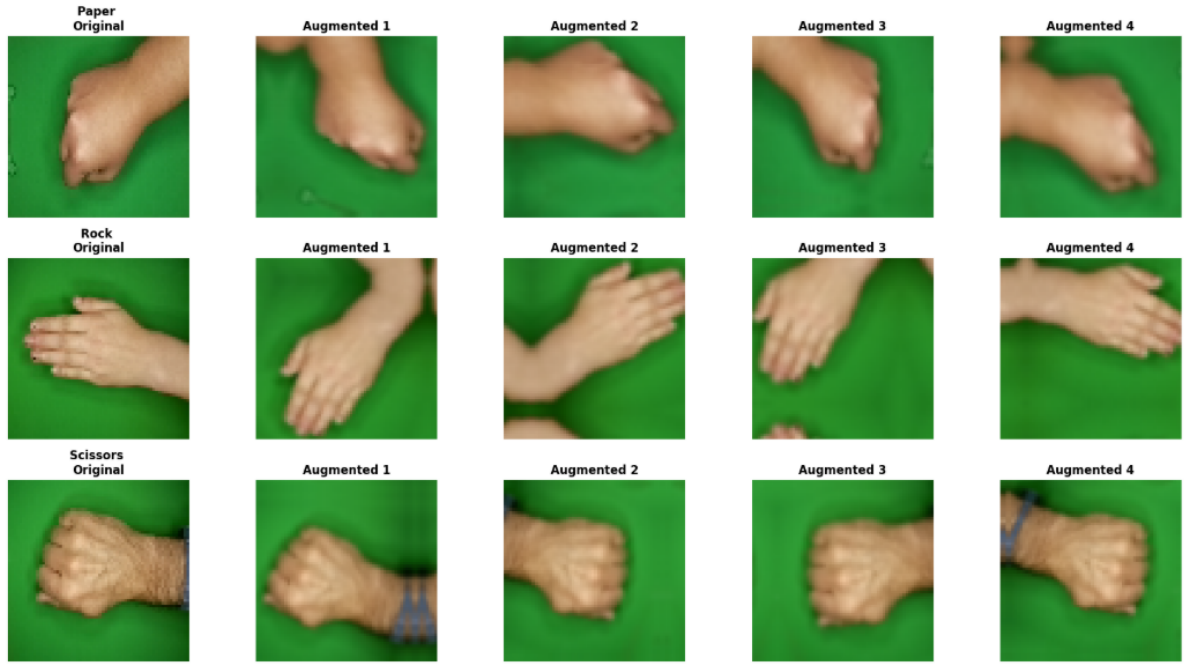

Figure 3: Examples of original and augmented images for each class.

The augmentation techniques applied to the images were the following:

- Random Horizontal Flipping
- Random Rotation
- Random Zoom
- Random Translation

These transformations were applied randomly at each training epoch, so the models would never see the exact same version of an image twice during training. This effectively increased the diversity of the training data without the need of collecting new images. The validation and test sets were kept completely unaltered throughout the whole process, since any modification to them would have compromised the reliability of the evaluation results.

5 Models implementation

In this part of the project, three CNN architectures of incremental complexity were implemented. Each model was built, compiled and trained independently to allow a fair comparison of their performance on the classification task. All the models were trained by using the Adam optimizer and sparse categorical crossentropy as the loss function, which is appropriate for integer encoded multi-class labels. The performance of all three models was first evaluated through the use of nested 5-fold cross validation, where the Intermediate model was additionally tuned using a fully automated grid search on the inner loop of the CV to find the best hyperparameter configuration. After cross-validation, all three models were trained, and for preventing overfitting and to restore the best weights found during training, an early stopping callback was applied.

5.1 Model 1: Baseline CNN

The first architecture built is the Baseline CNN which was designed as the simplest possible architecture to be the performance reference point for the other two models. The model consists of two convolutional blocks followed by a fully connected classifier:

- CONVOLUTIONAL BLOCK 1: The first block applies 32 filters of size 3×3 with ReLU activation to learn basic features such as simple textures or edges, followed by a 2×2 MaxPooling layer to reduce the spatial dimensions of the feature maps.
- CONVOLUTIONAL BLOCK 2: The second block applies 64 filters with the same configuration to capture slightly more complex features and patterns.

The feature maps are then flattened and passed through a Dense layer of 64 units with ReLU activation, before the final output layer with three neurons and softmax activation for the three-class classification. In this case, no dropout or any other regularization technique was applied, keeping the architecture as simple as possible. The model was then compiled with the Adam optimizer at a learning rate of 0.001 and trained for up to 20 epochs with early stopping with a patience of 5 epochs.

5.2 Model 2: Intermediate CNN

The Intermediate CNN extends the Baseline model by adding a third convolutional block to capture more complex feature hierarchies. In this case the architecture structure is the following:

- CONVOLUTIONAL BLOCK 1: The first block, like before, consists of 32 filters of size 3×3 with ReLU activation to learn basic features such as edges and simple textures, followed by a 2×2 MaxPooling layer.
- CONVOLUTIONAL BLOCK 2: The second block doubles the filters to 64 to learn more complex features such as combinations of edges and hand shapes, again followed by a MaxPooling layer.
- CONVOLUTIONAL BLOCK 3: In this case we added a third block which increased the filters to 128 to capture high level representations and details like specific hand gestures.

The feature maps are then flattened and passed through a Dense layer of 128 units with ReLU activation. A dropout layer in this case was added, of rate 0.5, to reduce the risk of overfitting by randomly deactivating neurons during training, helping the model generalize better to unseen data. The final output layer has 3 neurons with softmax activation. The model was then compiled like model 1 with the Adam optimizer at a learning rate of 0.001. During cross validation, 10 epochs per fold were used, while the final training was carried out for up to 20 epochs, both with early stopping with a patience of 10 epochs.

5.3 Model 3: Advanced CNN

The Advanced CNN implements a modern architecture following best practices inspired by networks such as ResNet and EfficientNet. It consists of four convolutional blocks with progressively expanding filter counts, followed by a classification head like for the other two previous models described above:

- CONVOLUTIONAL BLOCK 1: 32 filters of size 3×3 with the same padding, Batch Normalization and ReLU activation for basic feature extraction such as edges and textures, followed by MaxPooling.
- CONVOLUTIONAL BLOCK 2: 64 filters with the same configuration to learn more complex feature combinations, followed by MaxPooling.
- CONVOLUTIONAL BLOCK 3: 128 filters to capture hand gestures shapes and specific patterns, followed again by MaxPooling.
- CONVOLUTIONAL BLOCK 4: The additional block used for the advanced model is composed by 256 filters to learn the most abstract representations of complete gestures, followed by MaxPooling and a Dropout layer of rate 0.3.

Each convolutional block uses `use_bias = False` since BatchNormalization provides its own bias term, making the explicit bias redundant. Batch Normalization was applied after each convolution to normalize the layer outputs to zero mean and unit variance, stabilizing training and allowing the model to converge more reliably. After the four convolutional blocks, the Flatten layer was replaced in this case with a GlobalAveragePooling layer which averages each feature map into a single value, significantly reducing the number of parameters and helping prevent overfitting. Then there's the classification head which

is composed by a Dense layer of 128 units with ReLU activation, followed by a second Dropout layer of rate 0.3, and the final output layer with 3 neurons and softmax activation. The model was then compiled like the other models with an Adam optimizer at a lower learning rate of 0.0001. The choice of lowering the learning rate was made because the presence of Batch Normalization layers makes the training more sensitive to large weight updates, so a smaller learning rate allows the model to update its weights more gradually and stably, avoiding instability during the Batch Normalization statistics stabilization phase. During cross-validation, for this model, 30 epochs per fold were used to allow sufficient BatchNorm warmup time, while the final training was carried out for up to 50 epochs with early stopping with a patience of 10 epochs.

5.4 Model 2 Tuned

In the end only the tuning of model 2 was performed for runtime purposes. To find the optimal configuration for the Intermediate model, a nested cross-validation strategy was adopted in combination with a fully automated grid search on the inner loop. For the outer loop, a 5-fold stratified k-fold cross-validation was used to evaluate all the three models, while the inner loop used a 2-fold cross-validation applied only to each outer fold's training data to search over 24 hyperparameter combinations for Model 2. The hyperparameter grid used consisted of:

- Learning rate: {0.001, 0.01}
- Dropout rate: {0.3, 0.5}
- Convolutional filters: {(32, 64, 128), (64, 128, 256)}
- Dense units: {64, 128, 256}

The configuration of hyperparameters that won the most across all the five outer folds was selected as the globally best configuration by majority vote. The best configuration found was the following:

- Learning rate: 0.001
- Dropout rate: 0.5
- Convolutional filters: (32, 64, 128)
- Dense units: 256

This configuration won in 2 out of 5 outer folds. Model 2 was then retrained from scratch using the best configuration on the combined training and validation data to maximize the amount of data available for the final training phase.

6 Results

In this section, the results of the project are presented and discussed. From the cross-validation comparison of all three models, the analysis then moves to the training curves to understand how each model learned over time. The evaluation of the test set was then performed on all three models to compare their final performance. Since in different runs Model 2 tuned reached almost Model 3 accuracy or even surpass it, Model 2 was selected as the best model. This is because it demonstrates that with hyperparameter tuning, even a simpler model can achieve a really high accuracy, without the need to create a more complex architecture. In the end, to verify the models generalization abilities, a generalization test was done on personally taken images of hand gestures of the Rock-Paper-Scissors game. This last step was also taken because we can see also how the models react on real-world images with different backgrounds and lighting exposure.

6.1 Cross-Validation Results

The nested cross-validation was performed on all three models to obtain a reliable and unbiased estimate of their generalization performance. As it can be seen from Figure 4, Model 2 (Intermediate) achieved the highest and most consistent performance across all the five outer folds for this run, followed by model 1 (Baseline) which also showed stable results. Model 3 (Advanced) on the other hand showed the lowest mean accuracy with significantly higher variance across folds. This suggests that the Advanced model is more sensitive to the specific data split used in each fold, likely due to the higher complexity of the architecture combined with the limited size of the dataset and the reduced number of epochs used during cross-validation.

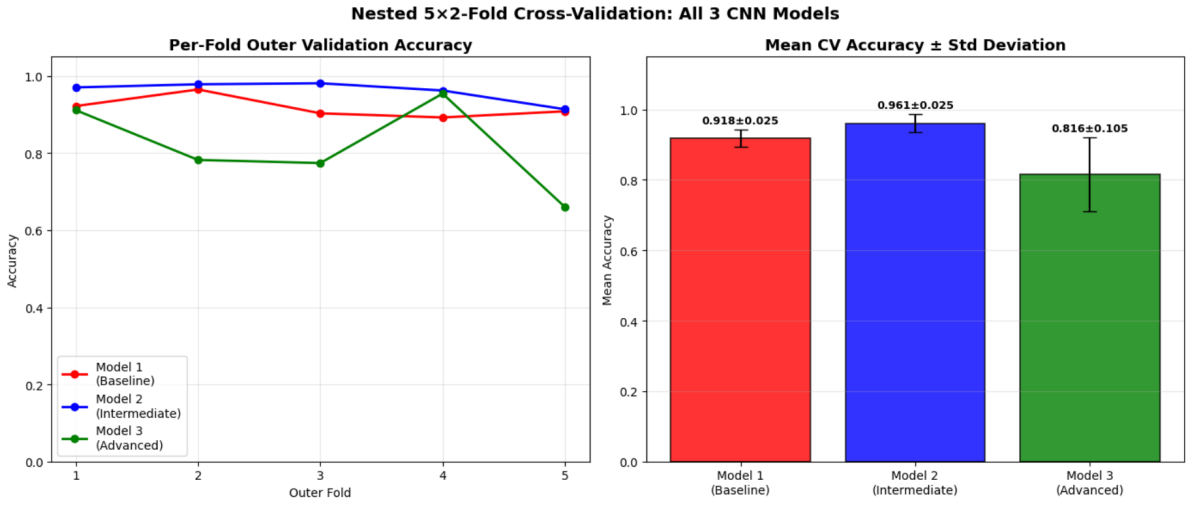

Figure 4: Nested 5-fold CV results for all three models

Regarding the hyperparameter tuning for Model 2, the best configuration was selected, as said before, by the majority vote across the five folds as shown in Figure 5. The folds highlighted in green represent the ones that selected the winning configuration which was then used for the final training of the tuned model 2.

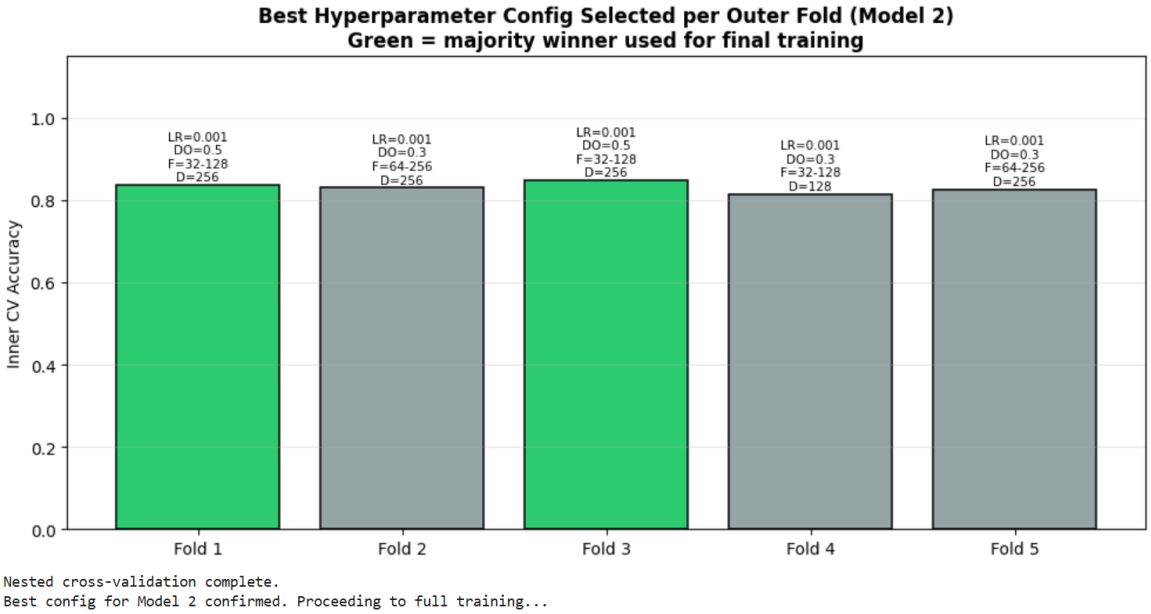

Figure 5: Best hyperparameter configuration selected per outer fold for Model 2.

6.2 Training Results

The training curves for all three models and the tuned version of Model 2 are shown in Figure 6 and Figure 7.

It is important to note that due to the stochastic nature of the training process, the results shown here refer to a specific run of the notebook. While the general trends and patterns described below were consistent across most runs, some runs produced different results, particularly for Model 3 which showed the highest variability between the runs.

For model 1 (baseline), both the loss and accuracy curves show a generally decreasing and increasing trend respectively, with some oscillations during training. An interesting observation is that the validation accuracy is consistently higher than the training accuracy throughout all the training process. This counterintuitive result is caused and expected by the use of the data augmentation in the processing of the data phase. This is because data augmentation introduced more variability to the dataset, making the training of the models harder, since the model sees augmented and more challenging versions of images during training, while validation set contains clean unmodified images. This confirms that the model generalizes well without memorizing the train data.

Model 2 (Intermediate) shows a similar pattern, with the validation accuracy again consistently higher than the training accuracy due to the effect of the data augmentation. The dropout layer further increased this gap while improving generalization. Both loss curves decrease smoothly and consistently towards the end of training, indicating stable learning behaviour.

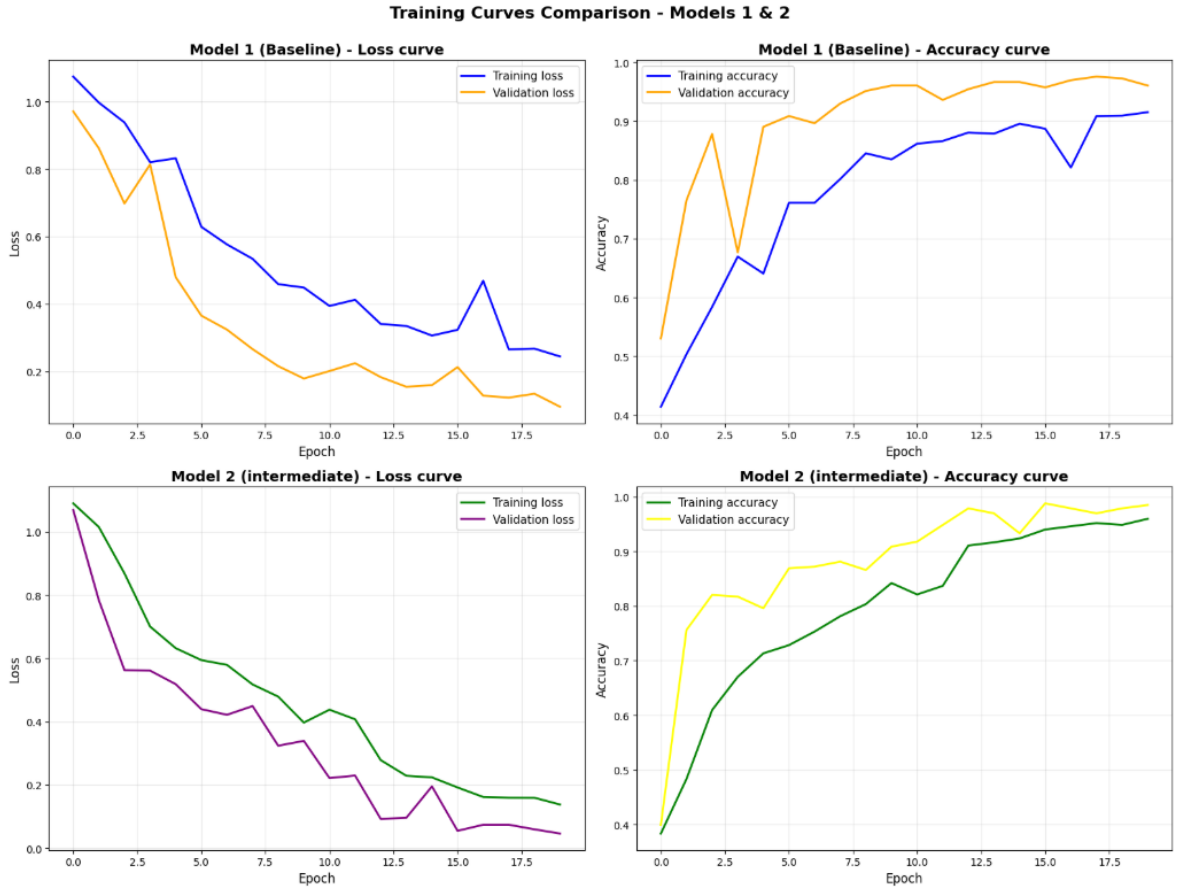

Figure 6: Training curves for Model 1 (Baseline) and Model 2 (Intermediate)

Model 3 (Advanced) shows a very different training behaviour in respect to the other two previous models. The validation loss initially increases significantly before dropping sharply around epoch 10, while the training loss decreases steadily from the beginning. This is caused by the BatchNormalization warmup phase, where the batch statistics need several epochs to stabilize before the model can start learning effectively. After this warmup phase the model reaches high accuracy, but the validation curves show more instability compared to the other two models.

In the end, for the tuned Model 2, since it was retrained on the combined training and validation data, only the training curves are available. Both the loss and accuracy curves show a smooth and consistent improvement throughout training, confirming that the model learned effectively from the larger dataset.

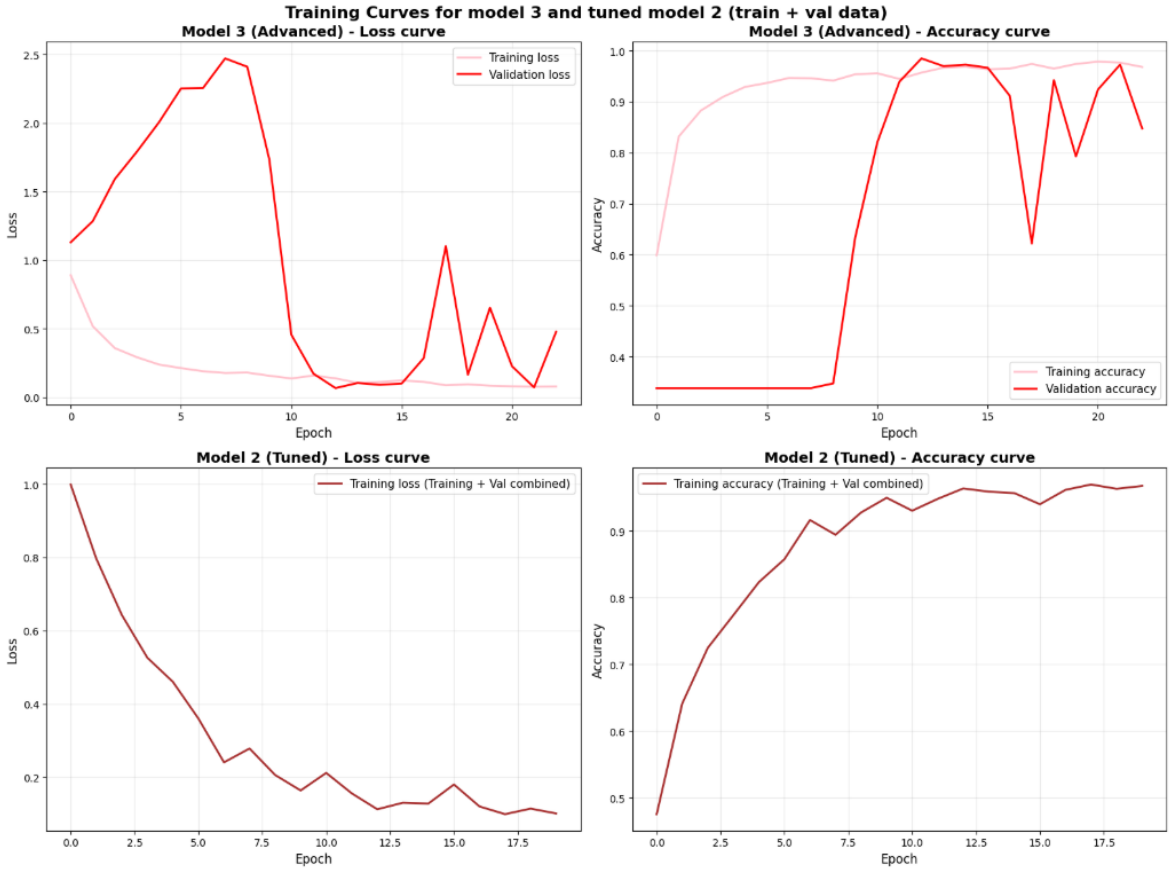

Figure 7: Training curves for Model 3 (Advanced) and Model 2 (Tuned)

6.3 Final evaluation on Test set

After the training, all the models were finally evaluated on the test set to compare their final performance. All models achieved in general high accuracy on the test set, confirming the fact that the architectures learned the classification task effectively. If we consider the untuned models alone, Model 3 (Advanced) achieved the highest test accuracy, followed by Model 2 (Intermediate) and Model 1 (Baseline). However, when comparing the tuned version of Model 2 with Model 3, the performance gap between the two is minimal, with model 3, in most of the runs, slightly outperforming the tuned model 2. This is a significant finding since it demonstrates that with proper hyperparameter tuning also a simpler model can perform almost the same or even outperform a more complex model. So, it means that with hyperparameter tuning great results can be achieved even without implementing the most difficult CNN architecture.

Since, Model 2 tuned, was decided to be the best model, an analysis of the misclassified images was conducted to see how many images exactly were misclassified and which type of images got misclassified, to see if there are some patterns in the images that we can identify as the cause of the misclassification.

First, a confusion matrix was created (see Figure 8). An interesting observation is that for the Scissors and Paper classes all the images were correctly classified, while the Rock class has 6 misclassified images.

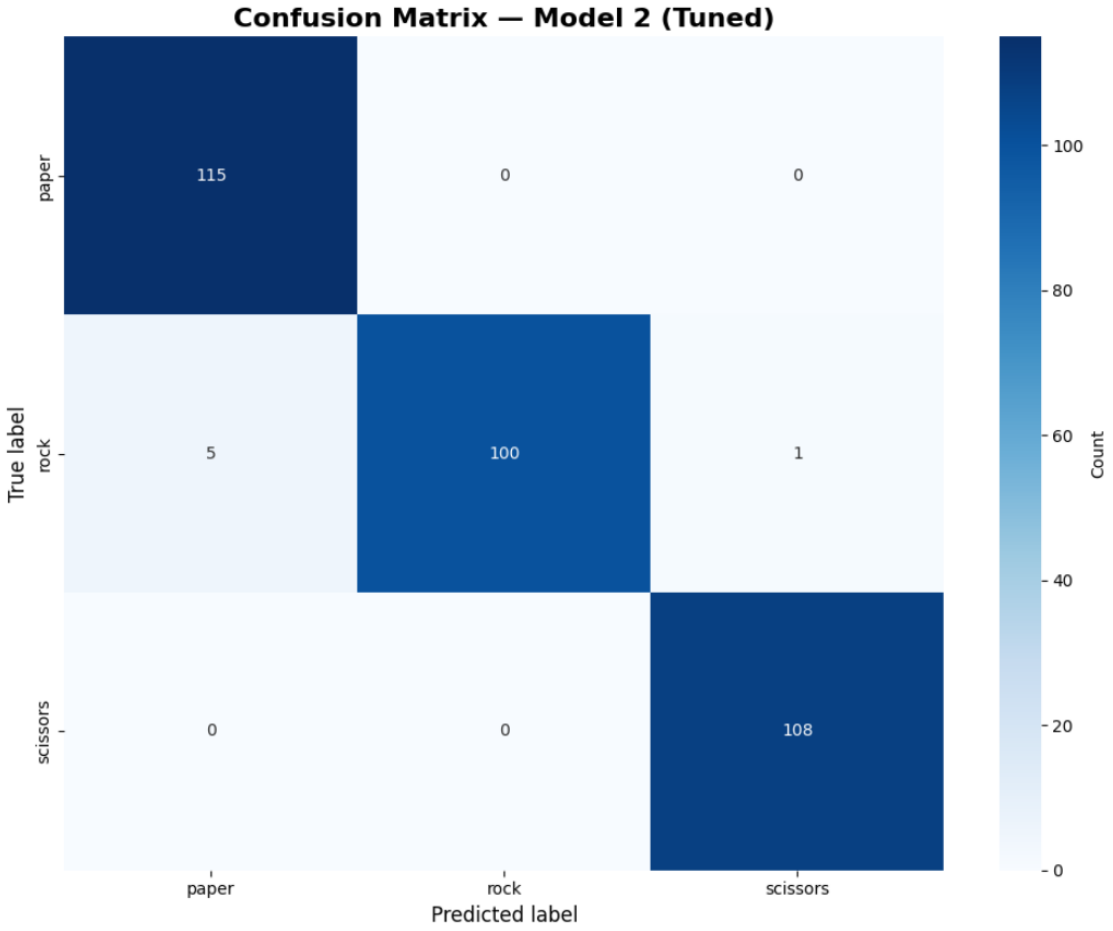

Figure 8: Confusion matrix for Model 2 Tuned

The misclassified images were then printed. Looking at them (see Appendix 17), this confusion is understandable since some Rock gestures appear quite flat and compact, making them visually similar to the Paper gesture. This suggests that the Rock class is the most challenging for the models to classify correctly.

6.4 Generalization test

To evaluate the generalization capabilities of the models on real-world images, a generalization test was performed using 45 personally taken photos (15 per class) of hand gestures of the Rock-Paper-Scissors game. The photos were taken under really different conditions such as a background with multiple colors and drawings and different lighting exposure. The results were striking: all four models achieved an overall accuracy of only 33%. All models classified the Paper class perfectly while achieving zero accuracy on both Rock and Scissors, predicting every single image of these two classes as Paper with extremely high confidence. This suggests that the models learned to rely on the green background of the original dataset images rather than on shapes and features of the hand gestures. This issue can be solved or improved through maybe background removal or more aggressive data augmentation techniques.

7 Conclusions

This project demonstrated that even with a small dataset, CNN architectures can achieve really high values of accuracy, provided that the methodology is rigorous and the architectures implemented are well suited for the problem chosen. Among all the three models considered, the Intermediate model (Model 2) was overall the best CNN architecture built because it achieved high accuracy even having a simpler architecture than model 3, achieving an accuracy similar or even higher, in some runs, to the accuracy of model 3 (Advanced). Model 3, even though it has a more complex architecture in respect to Model 2, during the cross validation phase it reached a lower performance in respect to Model 2 (Intermediate) and Model 1 (Baseline), this due to its higher complexity relative to the dataset size. This suggests that for small datasets, like the one used for this project, a moderate complexity architecture is the best choice because it tends to generalize better than deeper CNNs.

Another important thing to notice is the effect that data augmentation had on the models. Model 1(Baseline) and Model 2 (Intermediate) had almost every time validation accuracy higher than the training accuracy. This counterintuitive result is attributed to the effect of data augmentation, which artificially makes training harder for models by increasing the variability of the images, exposing the models to modified versions of the original ones, while the validation set remains unmodified and clean with no data augmentation. This is actually a good result because it demonstrates that both models generalize well to unseen data without memorizing the training data. Model 3 (Advanced), on the other hand, has a completely different behaviour in most of the runs. For this model, actually, the training accuracy is higher than the validation accuracy, this probably due to the BatchNormalization warmup phase, where the model needed several epochs to stabilize before learning effectively.

The nested cross-validation combined with automated grid search proved that with hyperparameter tuning, even a simpler model can reach the performances of a more complex one. However, this approach has a high computational cost and may not scale well to larger datasets or deeper architectures, where methods like Bayesian optimization or Random search could be more efficient alternatives to hyperparameter tuning.

In the end, the generalization test done with personally taken photos of rock-paper-scissors hand gestures, revealed that despite high test accuracy, all models failed to generalize in a real-world context with images with different backgrounds and different lighting exposures. This reveals a big limitation of the original dataset, which permitted the models to rely on the uniform green screen background instead of learning the actual hand gestures shapes and characteristics. Future improvements could address this through background removal or having a more diverse data collection

References

[1] Freeman, D. Rock, Paper, Scissors Dataset. Kaggle. Retrieved from: https://www.kaggle.com/datasets/drgfreeman/rockpaperscissors

[2] Alzubaidi, L., et al. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data*, 8(1), 53.

[3] Gupta, J., Pathak, S., & Kumar, G. (2022). Deep learning (CNN) and transfer learning: a review. *Journal of Physics: Conference Series*. IOP Publishing.

[4] Elngar, A. A., et al. (2021). Image classification based on CNN: a survey. *Journal of Cybersecurity and Information Management*, 6(1), 18–50.

[5] Chaganti, S. Y., et al. (2020). Image Classification using SVM and CNN. *2020 International Conference on Computer Science, Engineering and Applications (ICCSEA)*. IEEE, 1–5.

[6] Lei, X., Pan, H., & Huang, X. (2019). A dilated CNN model for image classification. *IEEE Access*, 7, 124087–124095.

[7] Sharma, A., & Phonsa, G. (2021). Image classification using CNN. *Proceedings of the International Conference on Innovative Computing & Communication (ICICC)*.

[8] Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems* (2nd ed.). O'Reilly Media. ISBN: 978-1-492-03264-9.

[9] IBM. Convolutional Neural Networks. Retrieved from: https://www.ibm.com/think/topics/convolutional-neural-networks

[10] GeeksForGeeks. Image Classifier using CNN. Retrieved from: https://www.geeksforgeeks.org/machine-learning/image-classifier-using-cnn/

[11] Dutta, S. Designing your own CNN model: a step-by-step guide. Retrieved from: https://medium.com/@sanjay_dutta/designing-your-own-convolutional-neural-network-cnn-model-a-step-by-step-guide-for-beginners-4e8b57836c81

[12] Self-Study Notes. What is CNN? Retrieved from: https://medium.com/self-study-notes/what-is-cnn-convolutional-neural-networks-4810e905af0d

[13] Data Science Stack Exchange. Exploratory Data Analysis with image dataset. Retrieved from: https://datascience.stackexchange.com/questions/29223/exploratory-data-analysis-with-image-datset

8 Appendix

Figure 9: Random sample of Paper class images.

Figure 10: Random sample of Rock class images.

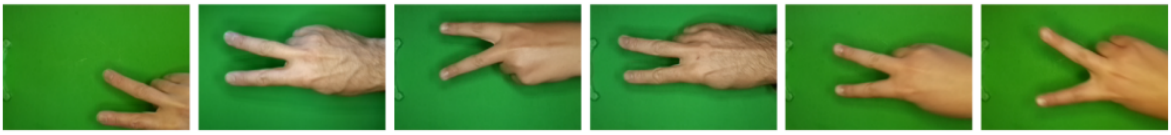

Figure 11: Random sample of Scissors class images.

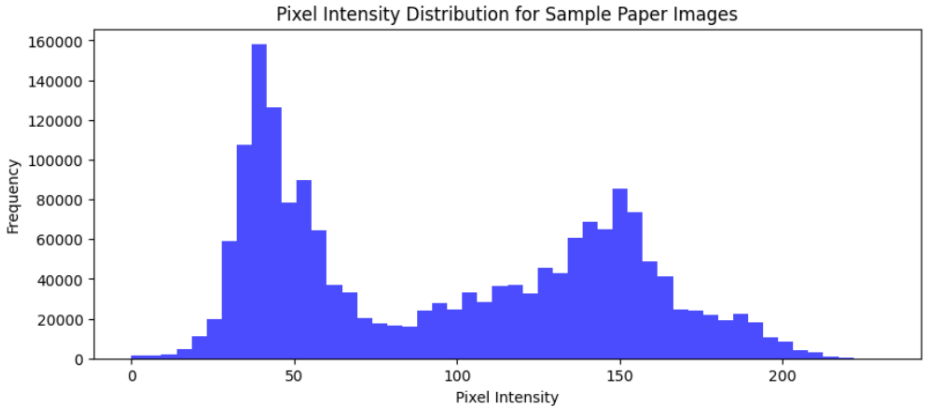

Figure 12: Pixel intensity distribution for Paper class.

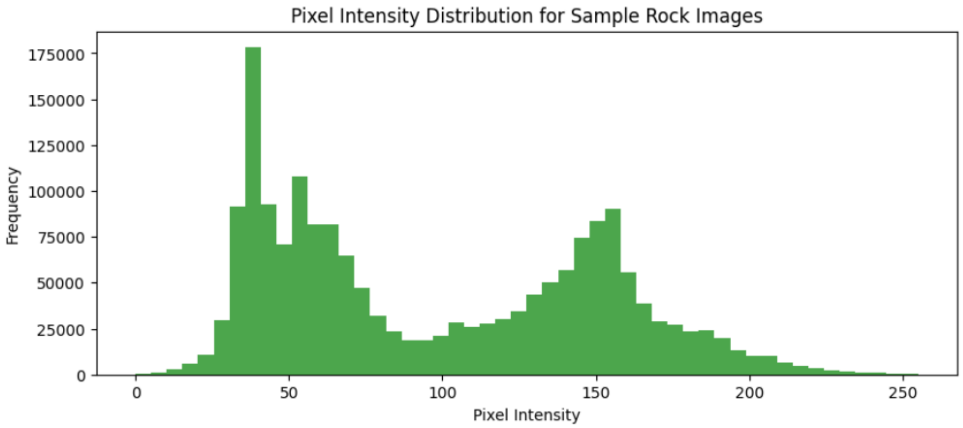

Figure 13: Pixel intensity distribution for Rock class.

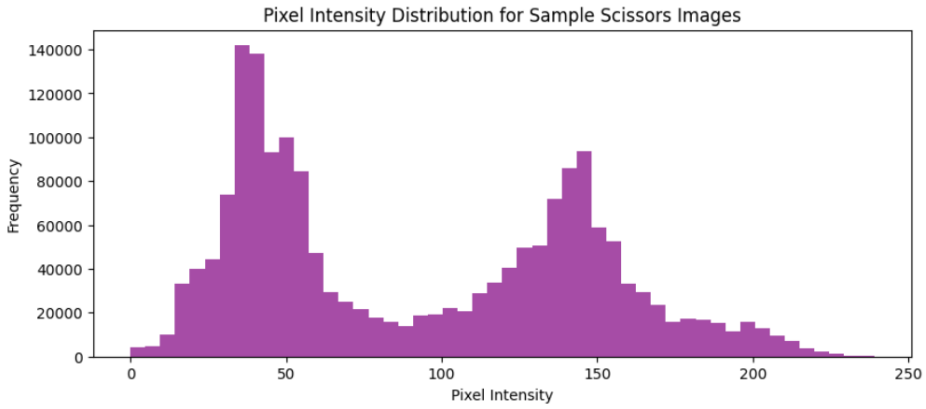

Figure 14: Pixel intensity distribution for Scissors class.

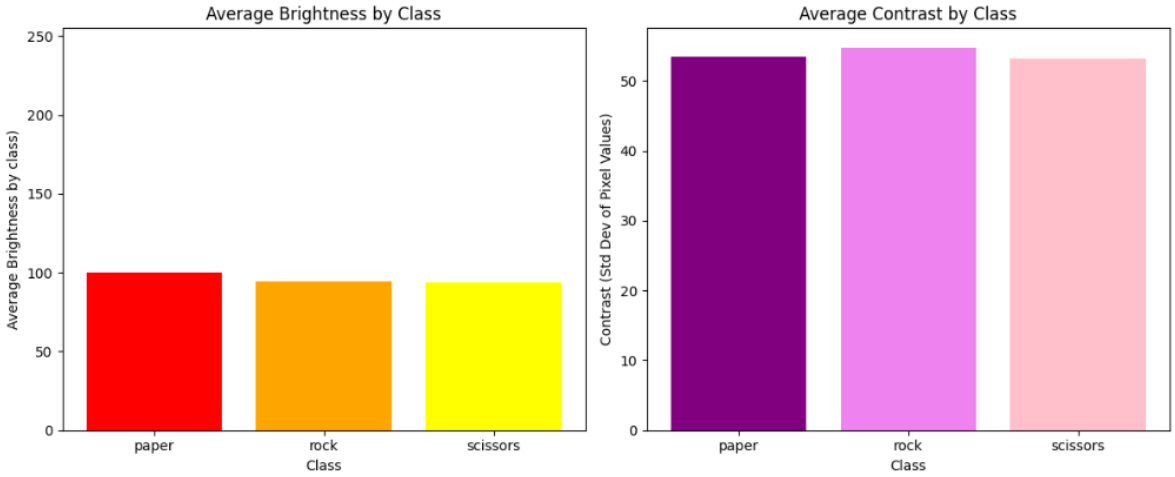

Figure 15: Average brightness and contrast values by class.

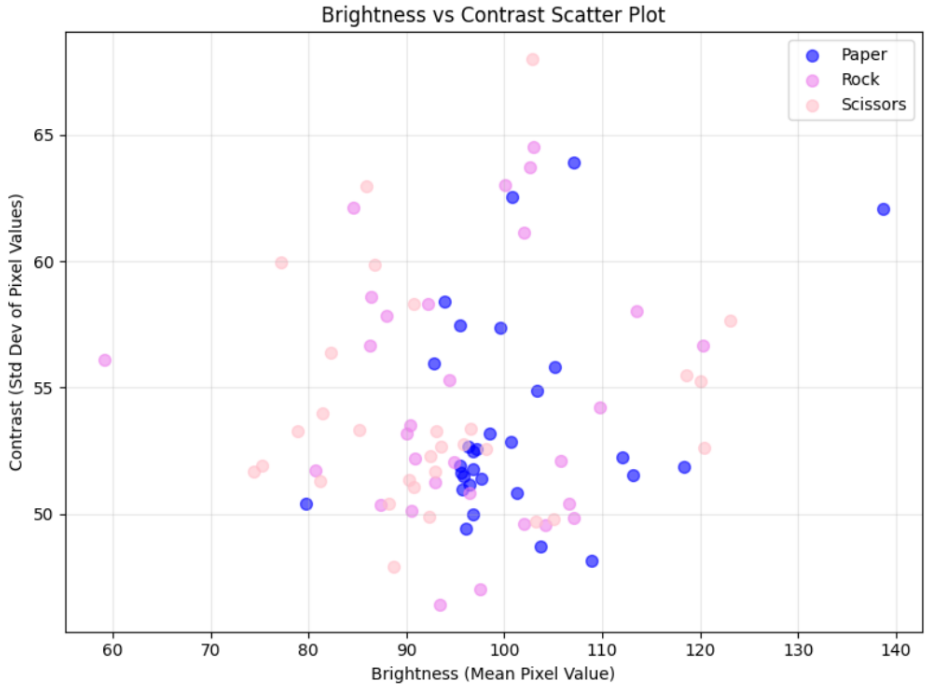

Figure 16: Brightness vs contrast scatter plot across all three classes.

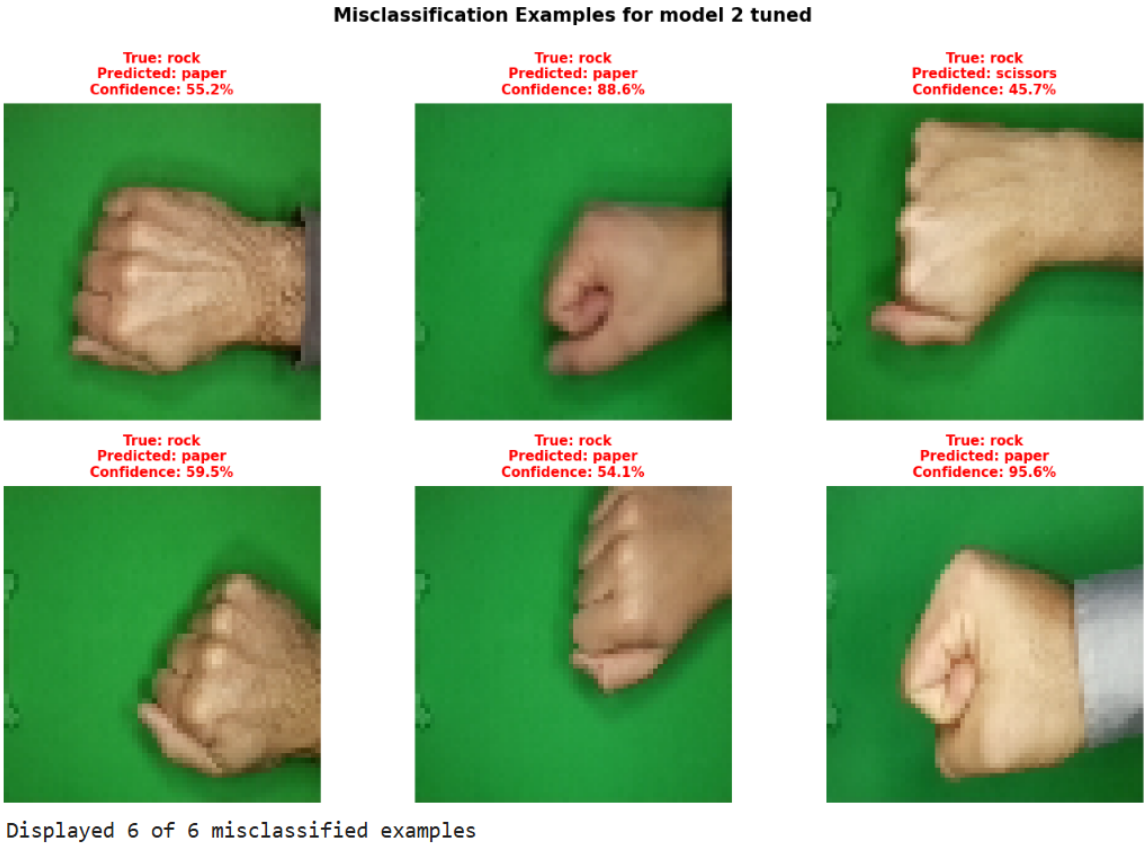

Figure 17: Misclassification images examples for Model 2 tuned